PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

AD

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 9/46, 9/38 — A1

(11) International Publication Number: **WO 00/23891**

(43) International Publication Date: 27 April 2000 (27.04.00)

(21) International Application Number: PCT/SE99/01904

(22) International Filing Date: 22 October 1999 (22.10.99)

(30) Priority Data:
9803632-0 22 October 1998 (22.10.98) SE

(71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON (publ) [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventors: LUNDSTRÖM, Lars-Erik; Fasanstigen 73, S-144 44 Rönninge (SE). TVEITE, Olav; Östgötagatan 89, S-116 64 Stockholm (SE). HINTUKAINEN, Kari; Skarpbrunnavägen 45A, S-145 64 Norsborg (SE). ISAKSSON, Nils; Kvambergsvägen 45A, S-141 45 Huddinge (SE).

(74) Agents: LINDÉN, Stefan et al.; Bergenstråhle & Lindvall AB, P.O. Box 17704, S-118 93 Stockholm (SE).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

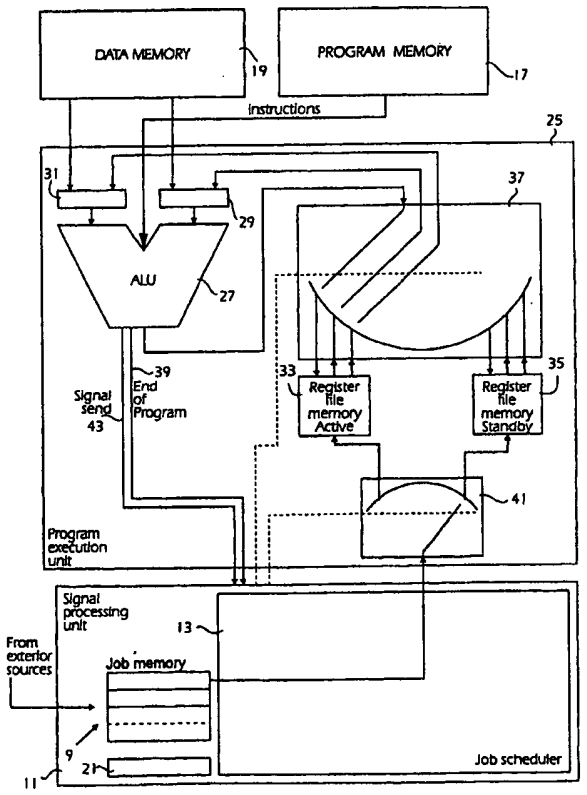
With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: A PROCESSOR

(57) Abstract

In a processor separate register file memories (33, 35) are arranged in a pipelined manner. The processor executes jobs having different priorities using for each job a register file stored in a register file memory assigned to the priority of the job. During the execution of the job data corresponding to a successive job of the same priority, the data of which for example is arranged ina queue (9), are stored in a standby register file memory (35). When finishing the ongoing job, the successive job having the same priority can start to be executed much faster since no or very little time is required for changing the contents of the register file memory. Instead, all that needs to be performed is to switch (37) to the standby register file memory to make it be connected to the Arithmetic Logic Unit ALU (27) of the processor. The previously active register file memory (33) will then be the standby one, and can receive a new register file for the same priority or a different priority when said successive job is executed. The processor is particularly useful for application in which jobs are frequently changed, such as a processor connected to a telephone exchange.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

A PROCESSOR

TECHNICAL FIELD

The present invention relates to a processor executing queued execution tasks or jobs and in particular to the handling of tasks or jobos and to a device for changing jobs in such a processor.

BACKGROUND OF THE INVENTION AND PRIOR ART

In telephone communication of today a computer is normally used at e.g. the connection place of subscriber lines. The computer handles the requests from the subscriber lines and when for example a subscriber wants to set up a connection to another subscriber line, it can send commands relating to the connection set up, debiting for connection time, etc. In such a computer the actual instruction processing will be made in a multitude of short execution sequences for each separate step or task to be completed. Such an execution sequence can be called a job. For normal traffic conditions a very large number of such short sequences must be executed at each instant. Then it naturally is very important that the execution of each job is done as fast as possible and also, that switches of or swapping between different jobs can be made very rapidly.

When a processor in a computer e.g. of the kind discussed above executes an execution task or job, it normally uses a number of registers for temporarily storing information, which is required later on in the execution, or information, which is to be output after completing the execution. In many computer systems of today, all such registers are joined together to form one logical unit, called a register file memory. The contents of the registers included in a register file memory is in the same way taken as a single logical unit, called a register file. During the execution of a job information can be output from the processor and then information in the register file memory is transferred to some external unit. After termination of a job the information stored in the register file memory is stored in some other memory and can be used as the output result of the execution or as a register file to be used when the same job is started again.

After terminating a job, a new job can be executed in the processor. Such a new job will most probably not use the same information which is currently stored in the register file memory and is the information left from a current or previous job. Therefore, the information in the registers of the register file memory in such a computer system must be changed when a new job is to be executed in the processor. The initial information required for execution of the new job is usually available in some buffer memory and is loaded into the register file memory before starting to execute the new job, i.e. the data stored in the register file memory have to be replaced when a new job is to be executed.

In applications such as telephone call processing described above, there are many jobs queued up waiting to be executed and the time for executing each job is relatively short or even very short. In such a case the time for changing the context or data in the

register file memory can become a substantial part of the time for executing a job.

Methods of reducing the time required for changing jobs have been disclosed in the prior art. Thus U.S. patent 4,367,530 discloses a control apparatus for an internal combustion engine. The control apparatus comprises an input/output unit including a first register file memory for storing constants and data produced by a central processor unit and a second register file memory for storing signals indicative of conditions of the engine.

Furthermore, U.S. patent 4,980,819 describes a two unit pipelined processor having a separate register file memory in each unit. The two pipelined processors are located on different chips and the register file memories of the two units are interconnected so as to share certain input data register stages to enable updating to take place within a minimum of time.

U.S. patent 5,357,617 discloses a hybrid pipelined processor for substantially concurrent processing of a plurality n of program instruction threads. The execution unit of the processor includes n sets of register file memories, each of which contains the working contents for a corresponding one of the plurality n of instruction threads.

In the published International patent application WO 98/36355 a processor using context-switching or more particularly context-cycling is disclosed. Each context is fixed and has fixed registers associated with it, different data being stored in the fixed registers. Furthermore, the processor context is switched after the execution of each instruction so that the same context cannot execute another instruction directly after an executed instruction. A special context, a timed context, is provided controlling the times when the other contexts are to execute instructions.

In the published European patent application 0 405 726 context switching is used in a processor system for two registers which can be simultaneously or individually connected to the processor. The contexts have no priority levels. In the published Japanese patent application 10207717 two register file memories are used, at each instant one of the register file memories being active and connected to the processor and the other one receiving data from the exterior or delivering data to the exterior.

SUMMARY OF THE INVENTION

It is an object of the present invention to overcome or at least reduce delay problems associated with changes of jobs in a processor.

It is a further object of the present invention to provide a computer system, which is capable of significantly reducing the execution time for applications involving frequent job changes.

The problem solved by the invention is thus how to make the changes of jobs in an electronic processing system as fast and seamless as possible and thus how to reduce the total processor time required for a job and thereby increase the efficiency of the processor.

Thus, in an electronic processor or electronic processing system separate, mutually independent, register file memories are used, each adapted to hold a register file to be used by the logic and calculation unit of the processor in executing jobs are provided. The jobs arrive to the processor and are processed and have one of at least two different priority levels. When the logic and calculation unit uses the register file stored in one of the register file memories, a second, separate, register file is read from or loaded to another one of the register file memories, in the lattter case holding the information required for the execution of the next job. So many register file memories are provided that one individual register file memory can at each instant be assigned to each priority level and that at least one extra, standby separate register file memory is provided which can be used for being loaded with a register file for job having the same priority level as that currently being executed.

Thus, when a job is finished and a next job having the same priority is to be executed, all that needs to be performed is to inform the processor that it shall use the information loaded into the standby register file memory when executing the next job, and so on. In such a system no or little time has to be spent on changing the information in the register file memory when changing jobs.

By using register file memories in such a "pipelined" manner, a considerable amount of processing time is gained. This is particularly true in the case where jobs are changed often, and where the jobs themselves are relatively short. This is for example true for a processor controlling a telephone switch or exchange. The extra register file memory is not primarily assigned to a definite priority level but can by a switching mechanism be made active and thus the register file memory for the priority level of the job to be executed. The former active register file memory will then be the standby one. This makes the switching, loading and unloading of register file memories efficient with a reasonable amount of extra hardware.

The separate, individual register file memories, as proposed in the prior art, are not intended for jobs having the same priority or jobs having to be executed in a sequential order. They are instead intended for jobs of different program threads or for programs processing very different types of data.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will now be described by way of non-limiting detailed embodiments with reference to the accompanying drawings, in which:

- Fig. 1 is a block diagram of a portion of a telephone network near a subscriber end,
- Fig. 2 is a schematic diagram of an electronic processor having two separate register files,
- Figs. 3a and 3b are diagrams illustrating the change of jobs and the time required therefor in the electronic processor of Fig. 2,
- Fig. 4 is a schematic diagram of an electronic processor executing jobs of two priority

levels,

- Fig. 5 is a diagram of a state machine for loading new jobs to an execution unit of the electronic processor of Fig. 4, and

- Fig. 6 is a diagram of a state machine for unloading jobs from an execution unit of the electronic processor of Fig. 4.

DESCRIPTION OF PREFERRED EMBODIMENTS

A portion of a telephone network is schematically illustrated in Fig. 1. A subscriber symbolized by the telephone set 1 is connected to the network through a line board 3, the line board having terminals for a plurality a subscriber connections. There is also a plurality of line boards 3. A group of line boards 3 is directly controlled by a regional processor 5 which senses the control signals from the subscriber set 3. The operation of the regional processor is controlled by a regional processor handler 7 which prepares information of jobs to be executed and also sends commands to the regional processor handlers. The information of the jobs are delivered to a queue called the job memory 9 in a signal processing unit 11. From the job memory 9 the job information is sent through the signal processing unit 11, as administered by a job scheduler 13, to an instruction processing unit 15 which comprises a general purpose processor or computer, here called program execution unit or instruction processing circuit 25, executing the jobs. The information of a job contains information on the program sequence to be executed and the data to be used during the execution. The program sequence, i.e. the sequence of instructions to be executed, are taken from a program memory 17 and the data to be used can be found in the information of the job but also in a data memory 19. In the processing of instructions new data can be produced and they are transferred to a regional processor 5 through the job scheduler 13 and an output register 21 in the signal processing unit 11. The regional processor 5 responds to received data and e.g. sets up a connection of a call through an exchange or switch 23 to another links and exchanges of the network and finally to another subscriber.

The structure of the signal processing unit 11 and the relevant part of the instruction processing unit 15, called the program execution unit 25, is illustrated in Fig. 2. The execution unit 25 is the central processing unit (CPU) 1 executing instructions as read sequentially from the program memory 17 and operating on data, such data e.g. being stored in the data memory 19. The job scheduler 13 contains a simple processor which operates according to a fixed microprogram and basically independently of or asynchronously of the processing made in the execution unit 25 and e.g. handles information associated with jobs in the job memory 9.

The part of the execution unit 25 actually executing the instructions is an arithmetic and logic unit, ALU 27 having in the conventional way two input registers 29, 31. Each of the input registers 29, 31 receives data from the instruction stream, from the data memory 19 and from other registers arranged inside the execution unit 25, these other

registers being collected in one of two register file memories 33, 35. The register file memories 33, 35 are arranged in parallel to each other. Only one of them is used by the ALU 27 at each instant as selected by a multiplexer 37 and from the selected register file memory thus data can be transferred to the input registers 29, 31.

The instructions to be executed by the ALU 27 are in the conventional way contained in or divided in longer sequences, called programs located in blocks in the program memory 17. A program can be started and then in the execution of the instructions contained therein use some input data obtained from the data memory 19 and also data from a register file, which thus when starting the execution of the program, is stored in the register file memory 33 or 35 used by the ALU 27 during the execution of exactly this program. For the start of a program a start address in the actual block is required, the start address indicating that instruction in the sequence of instructions in the program, which will be executed first. The same program can in the general case be started using different register files and different start addresses.

The system as illustrated in Fig. 2 receives commands or signals from the regional processor handler 7, see Fig. 1, informing on the programs to be executed together with some input data to be used by the execution unit. Each such command or signal requiring the program execution unit 25 to execute some special task is said to start or set up a job or task and therefor it stores in the job memory 9 information of the program to be executed, the start address of the program block, some job identification number and data to be used as a register file. For a new job thus the information of the job is written in the job memory 9 at the first empty position in the queue, the job memory thus holding a queue being the FIFO type.

When a job is terminated the ALU 27 issues a signal indicating this state on a line 39 to the job scheduler 13. If there is another job in the queue in the job memory 9 it will next be executed. The job scheduler 35 has then already loaded the register file of the next job in that one of the two register file memories 33, 35 which is in not active and is called the standby register file memory. The job scheduler then sends a signal changing the switch 37 to the former standby register file memory, which is already loaded with the register file for the new job. For the loading of the register file the job scheduler 13 controls the setting of another switch 41 in the program execution unit 25. This loading was then made during the processing of the former job, this procedure saving time during the switching of jobs. The job scheduler 13 unloads the used register file containing processed data from the former active register file memory into a memory 21 from which the data are transferred to the regional processor handler and the regional processor. A program can also send data during the execution thereof. Then it issues a signal on a line 43 to the job scheduler which then copies the contents of the active register file memory to the output memory 21 where it is accessed by the regional processor handler.

The saving of time associated with using an additional or extra "standby" register file memory is illustrated by the diagrams of Figs. 3a and 3b. In Fig. 3a the case of having only one register file memory is illustrated. A small time period is always required between two jobs and thus the process of Fig. 3a starts with such a period, called "Switching procedure between jobs". Then there is a time period for loading the register file, here called "Read data of a new job". Thereupon the relevant program can be executed, "Execution of a job". The job is terminated and then there is a time period required for unloading the register file, "Read data from register file". Then there is a new period for starting a new job, "Switching procedure between jobs". The procedure then continues in the same way.

In Fig. 3b instead, the loading and unloading of data is made during program execution, i.e. the time periods "Read data of a new job" and "Read data from register file" occur during the program execution periods, "Execution of a job". For the case of having short program sequences each requiring a short time, there will obviously be significant time saved compared to the case depicted in Fig. 3a. The saving of time will obviously be high if the time needed for loading and unloading jobs is significant in relation to the time required for processing of each job, but will be smaller if the loading and unloading times are small.

In the general case a system of the kind as described herein uses different priority levels. Thus a job having a higher priority level will always be executed before a job having a lower priority level. In Fig. 4 a processing system using two priority levels is schematically illustrated, jobs having priority level 1 being processed before jobs having priority level 2. Jobs having a lower priority may be interrupted in the case where a job having a higher priority is to be executed. The signal processing unit 11 and the program execution unit 25 are then modified. The job memory 9 holds two queues, one queue for jobs of priority level 1 and one queue for jobs of priority level 2, each queue being the FIFO type as above. The queuing of jobs incoming to the signal processing unit 11 from the regional processor handlers 7 and of other possible jobs to be placed in the job memory 9 are handled by a queue handler 10 taking out the priority of a job and placing the job at the end of the respective queue. Three register file memories 33, 34, 35 are arranged operating in parallel to each other, one register file memory always holding the active register file used by the program currently being executed. Another register file memory contains the job information of the next job having a priority level different from that which is currently executed. A third one of the register file memories is the standby one, which can be loaded and unloaded during the execution. The assignment of the register file memories to for instance the priority levels will then change when processing a sequence of jobs. The switch 37 is modified to be a 1:3 switch, so that the ALU 27 can use the selected one of the register file memories 33, 34, 35. Two parallel switches 41, 41 are arranged for transferring job data from and to the register file

memories, each such switch also being a 1:3 switch.

The switching or swapping between programs or jobs will in this case be much more complicated. For instance, as has already been mentioned, a program which has a low priority level and is executed can be temporarily stopped when another program being more urgent or important has to be executed, this other program then having a higher priority level.

A change of the job which is being executed occurs in following cases:

- The job is terminated or finished in a regular way. Such a termination of a job may be arranged by entering a special instruction in the sequence of instructions of the program being executed, such an instruction for example being called End of Program, EP. When executing this instruction, the ALU 27 issues some signal indicating this state to the job scheduler 13. If there is another job in the queue in the job memory 9 storing the jobs belonging to the same priority class it will next be executed and then a switch to the standby register file memory 33 - 35 is made. Otherwise the first job stored in the queue for the next lower priority class will be executed.

- A job belonging to a higher priority class is entered from the exterior in the job buffer and its register file has been loaded to the current standby register file memory 33 - 35. The job scheduler then issues an interrupt signal to the ALU 27 on a line 45. The instruction sequence being executed is then stopped at the next possible breaking point and the ALU sets the switch 37 to the standby register file memory.

The job scheduler 13 loads and unloads the register file memories 33 - 35. The change of active register file memory is performed by the ALU 27 as symbolized by the block 47 "Register file memory control".

The first steps of setting up a connection from the subscriber's telephone set 1 will now be described. First a subscriber is assumed to lift the handset of the telephone 1 generating a hook-off signal in the line board 3. This signal is transmitted through the regional processor 5 and the regional processor handler 7 to form an information packet or job, such a job containing Information and Header fields and a data field. The Information and Header fields contain information on the event occurred, i.e. hook-off in this case. The data field contains identification data of the input terminal of the line board 3, to which the calling subscriber is connected, and of the respective line board 3, regional processor 5 and regional processor handler 7. Furthermore, the Information and Header fields have information for example setting the priority of the job to priority level 2. Directly after setting up this job the regional processor handler 7 normally sets up another job associated with some new event, such as detecting that the telephone set of another subscriber has entered the state of hook-off, that numbers have been dialled at some telephone set or that some telephone set has gone into the state of hook-on.

The job associated with a new call to be set up as described above is received by the queue handler 10 and placed in the last position in the portion of the job memory

arranged for jobs of priority level 2. After some time no more jobs of priority level 1 are left and possibly after some more time this job will be the first in the queue for priority level 2. The job scheduler 13 then first checks the contents of a register 51 "Trying to take new job" in the program execution unit 25 for an indication that a new job can be loaded. It waits until the indication is positive, and then it checks that there is a register file memory 33 - 35 which can receive a new job. If there is no such memory, it waits until there is one. Then the program scheduler accesses the job in the job memory, extracts the contents of the fields Information and Header and places them in a memory 53 called "Info-Header In" where they are available to the ALU 27. Finally it extracts the data of the job from the data field thereof and places them in that one of the register file memories 33 - 35 which is ready to receive a register file for a new job, this register file memory being called the standby one. After finishing the loading of the register file an indicator is set in a register 55 "New job" telling that there is a new job which is ready to be executed and that its register file is loaded.

When in the instruction flow executed by the ALU 27 next an instruction is found signalling the end of the program the ALU performs a sequence of steps found in the micro instructions corresponding to said instruction. These steps are symbolized by the block 47, "Register file memory control". Then first the indicator stored in the block 51, "Trying to take new job" is set to a state signalling to the job scheduler 13 that no loading or unloading of register file memories must now on be performed. Then the contents of the "Info-Header In" memory 53 is accessed and in particular the event identification. By a table look-up the actual program block and start address are found and transmitted to the device handling the instruction flow, not shown. The priority level of the new job is copied to a register 57 "Priority level" in the program execution unit 25. Then the swap of the switch 37 to the standby register file memory is made, the number of this memory being found in a list 61. Then the rest of the contents of the memory 53 "Info-Header In" of the new job is checked. Finally, the contents of some of the registers are changed, i.e. the contents of register 55 "New job" is set to indicate that there is no more job ready, the contents of a register 59, "Job taken" is set to indicate that the job the register file of which is in the standby register file memory is now started to be executed, the contents of the register 51 "Trying to take new job" are changed to indicate that there is now activity of the ALU 27 associated with the start of the new job and thus the job scheduler 11 can load or unload the register file memories 33 - 35 as required. The microinstructions of the block 47 are then terminated and the ALU 27 continues to execute the instructions as taken from the program block using the start address as found earlier.

The program now executed will first identify the subscriber by using the data of the register file. It is then checked that the subscriber is allowed to make a call. The next expected event is receiving the numbers of a called another subscriber. Therefor the

program is first terminated and simultaneously a new job is made which is to prepare for the reception of dialled numbers.

In the termination process thus a new job is set up. A sequence of microinstructions initiated by a single program instruction is performed. First it is checked whether the job scheduler 11 has already used the information in a memory area 63 called "Info-Header Out" to send the result of a former job to either the output register 21 or to put a new job in the job memory 9. The job scheduler 11 sets a state indicator for such a sending operation in a register 65 called "Busy". If the contents of the register "Busy" indicate that the information has been used, information and header for the new job are stored in the memory 63 "Info-Header Out". This information contains an indication of the program to be started, i.e. its block number and its start address, and the priority level, which is one step higher than the present job being terminated, i.e. priority level 1. Information of the new program to be started is found by looking up in a table stored in the data memory 19, the input to the table being e.g. a number of the program currently being executed. In the register file memory for priority level 2, the memory 35 as illustrated in Fig. 4, there is still stored the data identifying the calling subscriber. Finally the state of an indicator stored in a register 67 "Job signal complete" is set to indicate to the job scheduler 11 that the memory 63 "Info-Header Out" and the register file memory for the present job contain information belonging to a new job or information to be sent to external devices, i.e. to the regional processor handler. Then the microinstruction sequence is finished and the ALU 27 will try to take a new job as described above.

The job scheduler 11 recognizes the state of indicator in the register 67 "Job signal complete", sets the indicator in the register 65 "Busy" to indicate the unloading state and starts to take care of the information in the memory area 63 "Info-Header Out". It decodes some of this information to find whether it is information directed to external devices or a new job. Here it finds that a new job will be started. It then accesses the priority level of the new job and loads the new job to the appropriate queue, in this case the queue for priority level 1, copies the contents of the memory 3 "Info-Header Out" and of the correct register file memory, here 35, to the job memory 9, in the last position of the queue. Finally the indicators in the registers 67 "Job signal complete" and 65 "Busy" are changed. This portion of the job scheduler 11, which can be an independently operating state machine, then starts testing the indicator in the register 67 "Job signal complete" whether there is new information to be received from the program execution unit 25.

The new job will in some time be executed by the ALU 27. It will then take the identifier of the subscriber slot from the associated register file and reserve a temporary memory area in the data memory 19. It places there data of the subscriber which can be required in the further processing of the call. The address of the temporary memory area

is also stored in the currently used program block in the program memory 17. This program generally also contains instructions for handling telephone numbers. The address of the temporary memory area is stored at a position in this block which is fixed for the subscriber connected to the terminal of the line board, to which the subscriber is connected. The program is then terminated and no new job is set up and no data is sent to external devices.

The subscriber then dials the number to which a connection is desired and then a new job will be created by the regional processor handler, this job containing the numbers dialled by the subscriber. This new job can have the lower priority and will be processed by instructions of the same block as the former job preparing for the number reception but using another start address. Then the address of the temporary memory area in the data memory 19 is taken from the memory position in the block corresponding to the number of the subscriber slot. The dialled number is checked for validity by using e.g. table look-ups in the data memory 19. An order for set-up of the call can then be given to the regional processor handler by sending data to the external buffer 21.

The different events and the corresponding jobs for a connection to be made and being made can be said to belong to the same "forlopp". The events and the processing of jobs belonging to one "forlopp" are independent of the events of the processing of jobs belonging to another "forlopp", considering that in the job processing some steps having a higher priority will always be executed before steps having a lower priority, independently of the "forlopp" to which they belong. Anyway, the events of different "forlopps" of course occur quite asynchronously of each other. The processing of jobs belonging to one "forlopp" can then be interleaved with jobs belonging to other "forlopps". The order of executing the jobs are determined only by the times when they are stored in the input queue 9 and by the priority of the jobs.

When executing instruction of a job having a lower priority the execution can be interrupted by a job having a higher priority. The ALU 27 will then receive an interrupt signal on the line 45 and save information in order to be capable of restarting the current job. Then it starts the procedure as above of trying to take new job. The job scheduler makes a new job of the interrupted one and in particular creates the fields Info and Header and places the job first in the respective queue, in this case in the queue for priority level 2. The information in the Info-Header fields then tells that an interrupted job has its register file stored in the register file memory for this priority level.

The register file memory control 47 uses the list in registers 61 and sets the contents thereof when changing jobs. Three registers are used, one for each priority level, i.e one register for priority 1 and one for priority 2, and a third register telling which register file memory is the standby one and holds the standby register file. The registers are updated when swapping to a new register file memory to be used by the

program execution unit.

The procedure executed by the job scheduler 11 in loading data for a new job to the standby register file memory is illustrated by the state machine diagram of Fig. 5. The state machine is mostly in an idle state 501. In this state the contents of the indicator in the register 69 "Job read out complete" is constantly checked whether it has been set. This indicator signals that all data and information from a former job have been read out to either the output buffer 21 or to form a new job stored in a queue in the job memory 9. When this indicator has been set, the state 503 is taken. In this state it is constantly checked whether there is a new job in the job memory 9. When a new job is detected, the state 505 is entered. There first the indicator in 67 just checked is reset and the indicator in register 65 "Busy" is set. The first job of the highest priority level is read and the respective fields are loaded to the memory 53 "Info-Header In" and to the standby register file memory as indicated by the indicator in 61. Then the state 507 is taken and there the indicator in 65 "Busy" is reset, the indicator in 55 "New job" is set. This state 507 is maintained until the indicator in 59 "Job taken" is set or a new job has been received by the job memory 9 having a priority higher than that of the job preloaded in 53 "Info-Header In" and the standby register file memory. In the first case the idling state 501 is taken again. In the second case the job having a higher priority is to be loaded. Thus a state 509 is entered in which the indicator in 65 "Busy" is set and the indicator 55 "New job" is reset. The fields of the new job are loaded as above in state 505. Finally the indicator in 55 "Busy" is reset and an interrupt signal is sent to the ALU 27 on line 45. Then the idling state 501 is entered again.

Another procedure executed by the job scheduler 11 is illustrated by the state machine diagram of Fig. 6. This procedure takes data from the standby register file memory and the memory area 63 "Info-Header Out" and transfers them to the output buffer. The state machine is mostly in an idling state 601. In this state the contents of the indicator in the register 67 "Job signal complete" is constantly checked whether it is set. When it is set, a state 603 is taken. Here the appropriate portion of the memory area 63 signalling whether the output information is a new job or data to be communicated to external devices. In the first case the state 605 is taken. There the priority of the new job is determined by reading the appropriate portion of the output memory 63 "Info-Header Out". Then the contents of the memory 63 and the standby register file are copied to the last place in the queue for this priority in the job memory 9. In the second case a state 607 is taken in which the contents of the standby register file memory are copied to the output buffer 21. Finally, after completing the operations in states 605 and 607 a state 609 is entered, in which the indicator in 67 "Job signal complete" is reset. After this state the idling state 601 is again taken.

Owing to the required high speed of processing the jobs, most of the functions illustrated in Figs. 2 and 4 are implemented in hardware, so that a minimum of register

polling and microprogram execution is performed. The system as depicted in Fig. 4 will in the general case handle a plurality of priority levels and also external interrupts requiring that jobs are executed immediately, as fast as their associated register file is loaded. Such external interrupts can have their own priority levels, which are all higher than the ordinary priority levels illustrated in Fig. 4. The operation of the job scheduler will then be much more complicated since there will be more stops of current executed programs and loading and reloading of the register file memories.

CLAIMS

1. A processor for processing jobs, which arrive to the processor, each of the jobs having one of at least two priority levels, a job of a higher priority level being executed before a job of a lower priority level and for each of the priority levels, jobs having the same priority level being executed in a sequential order which is the same as the order, in which the jobs having the same priority level arrive, the processor comprising a logic and calculation unit executing the jobs, **characterized by**

a plurality of register file memories, at each instant one individual register file memory being assigned to each priority level and in addition at least one register file memory assigned to be a standby register file memory, the register file memories each being adapted to store a register file which belongs to a job and is to be used and/or is used and/or has been used by the logic and calculation unit when executing the job, the register file memory assigned to a priority level storing a register file for a job having the priority level and the at least one register file memory assigned to be a standby register file memory storing a register file of a job which has been executed or is to be executed,

means connected to the processor and the register file memories for selecting for a job to be next executed, in the case where the job to be next executed and the currently executed job have the same priority level, the at least one standby register file memory as a register file memory to be used by the next job to be executed, and in the case where the job to be next executed has a lower priority level than the currently executed job one of the register file memory for said lower priority level and the at least one standby register file memory to be used by the job to be next executed,

means connected to the register file memories for accessing, during the execution of a job, a register file containing data belonging to the job, which is to be next executed after the job currently being executed, and storing the data in a respective one of the register file memories including the at least one standby register file memory which is not used by the job being currently executed.

2. A processor according to claim 1, **characterized in** that the means for selecting include control means for controlling a switch connected to the register file memories including the standby register file memory for selecting one thereof as a register file memory used by the job currently being executed.

3. A device for changing a job in a processor, the jobs having at least two priority levels, the processor comprising

a logic and calculation unit for executing the jobs,

register file memories each adapted to store a register file to be used by the logic and calculation unit when executing a job,

a control unit connected to the logic and calculation unit and the register file memories,

characterized in that

such a number of the register file memories are provided that at each instant they include one register file memory assigned to each priority level and in addition at least one register file memory assigned to be a standby register file memory,

the control unit including selection means for selecting, when the execution of a job is finished and the logic and calculation unit is to start executing a new job, one of the register file memory assigned to the priority level of the new job and the at least one register file memory assigned to be the standby register file memory to be connected to and used by the logic and calculation unit in the execution.

4. A device according to claim 3, **characterized by** means connected to the register file memories for accessing, during the execution of a job, a register file containing data belonging to a new job, which is to be executed directly after the job being executed, and storing the data in one of the register file memory assigned to the priority level of the new job and the at least one register file memory assigned to be the standby register file memory, the accessed register file memory being one in which the register file of the job being executed is not stored.

5. A device according to claim 4, **characterized in** that the means for accessing are arranged to store the data in the at least one register file memory assigned to be the standby register file memory in the case where the new job has the same priority as the job being executed.

Fig. 1

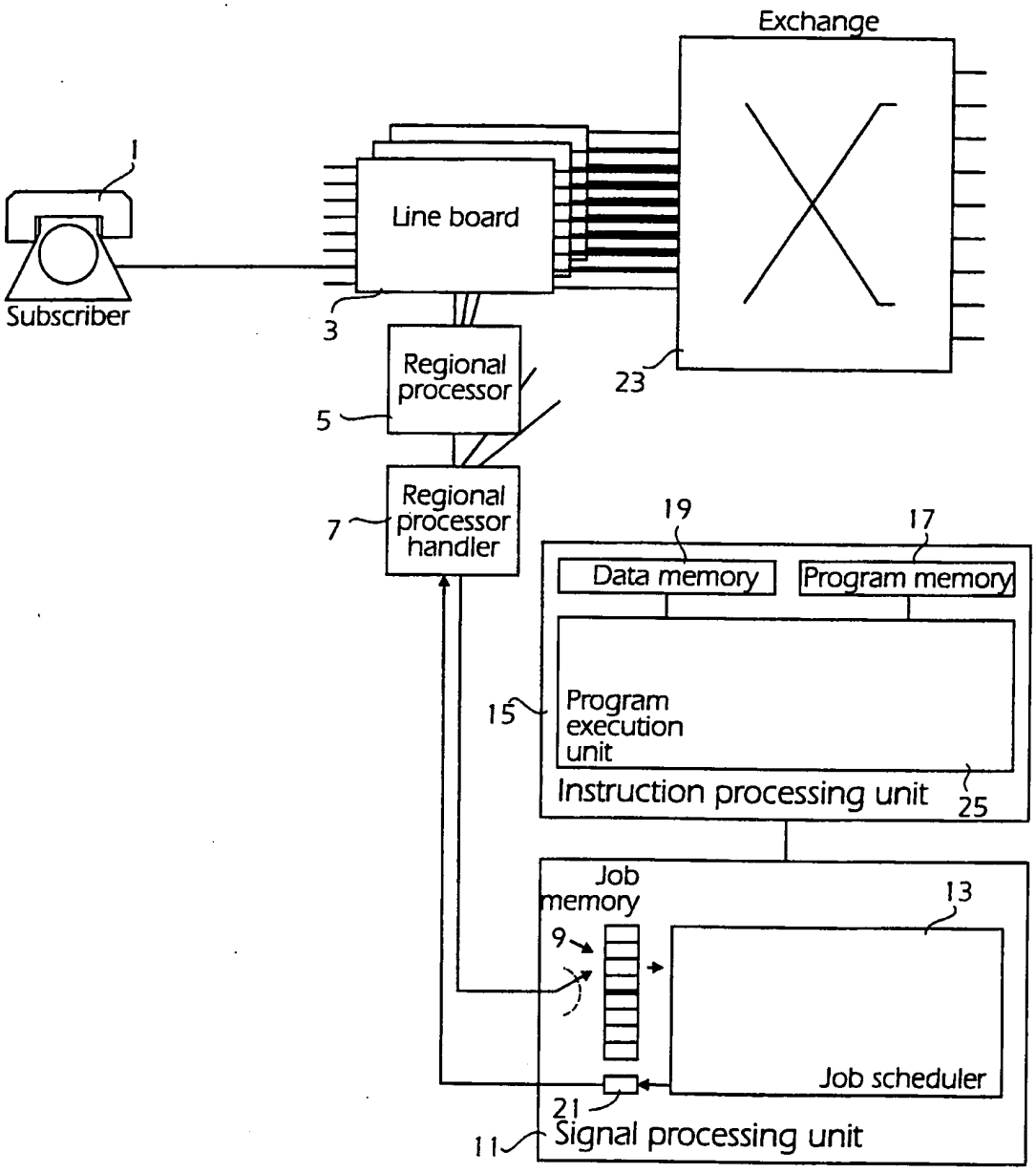
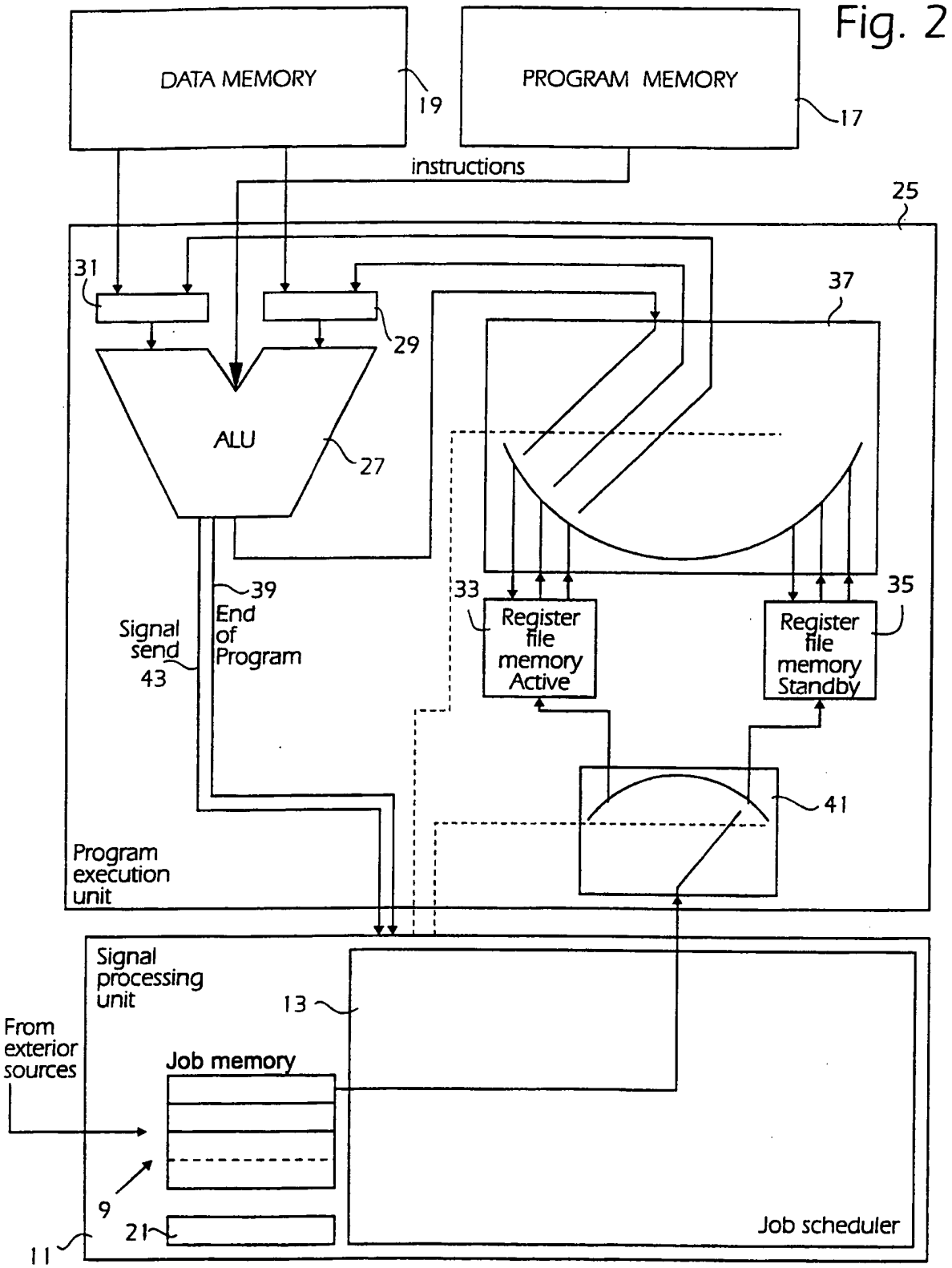

Exchange
1
Subscriber
Line board
3
23
Regional processor
5
Regional processor handler
7
19
17
Data memory
Program memory
15
Program execution unit
Instruction processing unit
25
Job memory
9
13
Job scheduler
21
11
Signal processing unit

Fig. 2


DATA MEMORY
19
PROGRAM MEMORY
17
instructions
25
31
29
37
ALU
27
39
End of Program
Signal send 43
33
Register file memory Active
35
Register file memory Standby
41
Program execution unit
Signal processing unit
13
From exterior sources
Job memory
9
21
11
Job scheduler

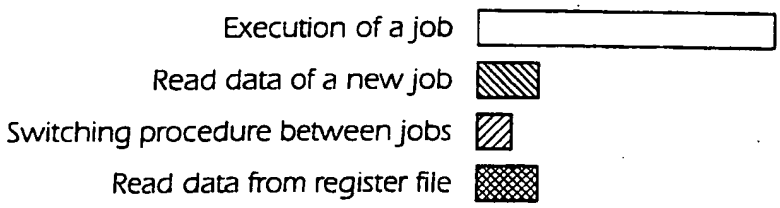


Fig. 3a Prior art

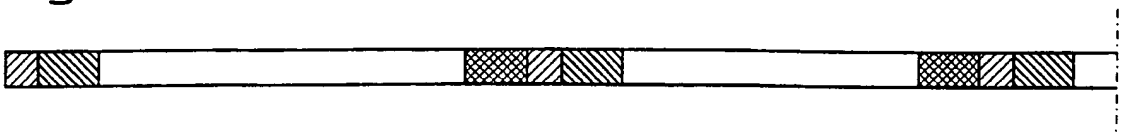


Fig. 3b

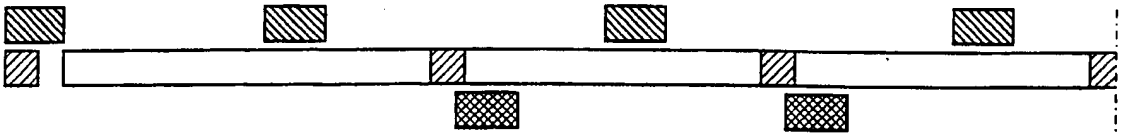


Fig. 4

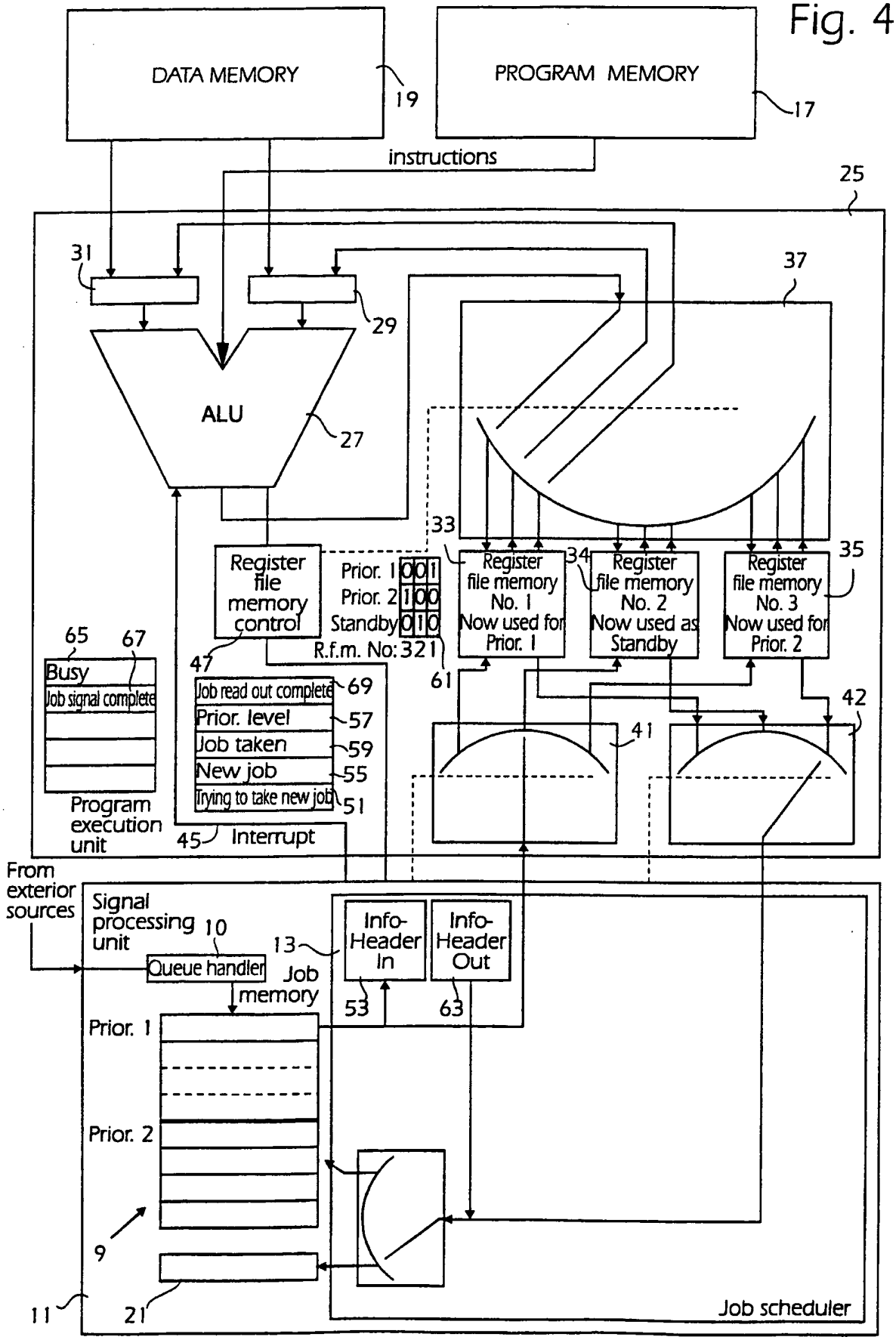

DATA MEMORY
19
PROGRAM MEMORY
17
instructions
25
31
37
29
ALU
27
33
34
35
Register file memory control
Prior. 1 0 0 1
Prior. 2 1 0 0
Standby 0 1 0
R.f.m. No: 3 2 1
Register file memory No. 1 Now used for Prior. 1
Register file memory No. 2 Now used as Standby
Register file memory No. 3 Now used for Prior. 2
65
67
Busy
Job signal complete
47
61
Job read out complete
69
Prior. level
57
Job taken
59
New job
55
Trying to take new job
51
42
41
Program execution unit
45
Interrupt
From exterior sources
Signal processing unit
10
Queue handler
13
Info-Header In
Info-Header Out
Job memory
53
63
Prior. 1
Prior. 2
9
11
21
Job scheduler

Fig. 5

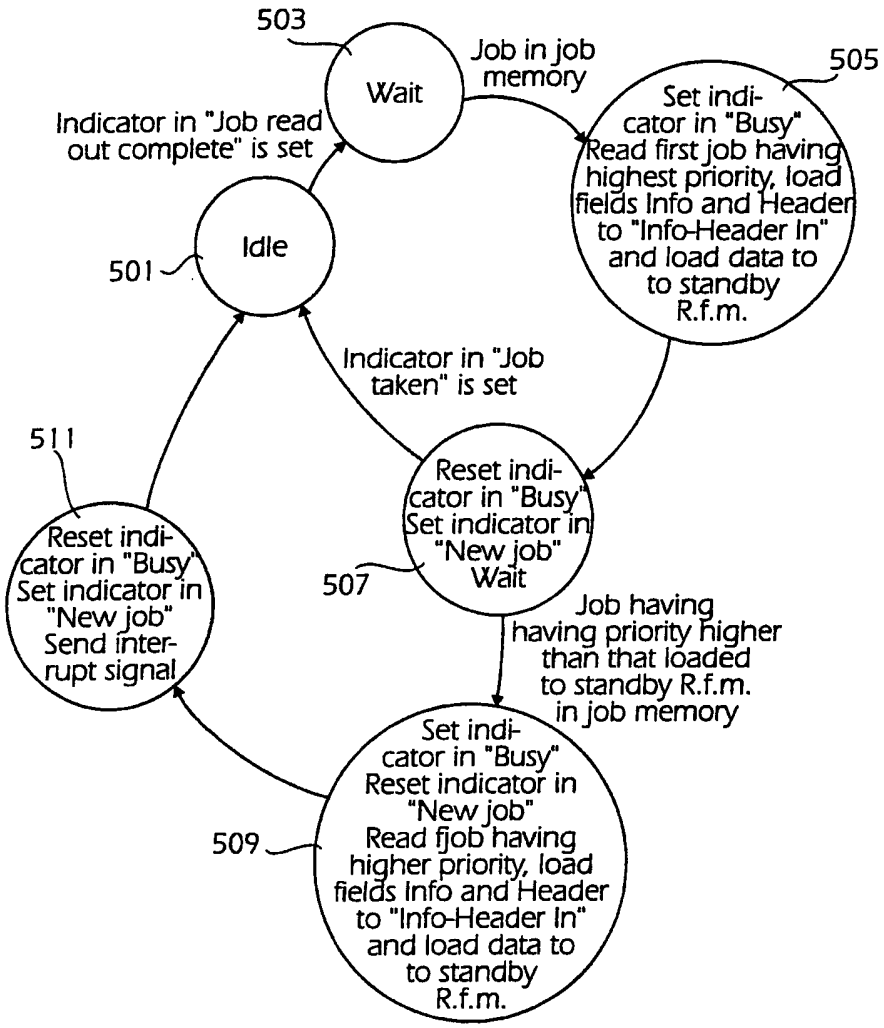
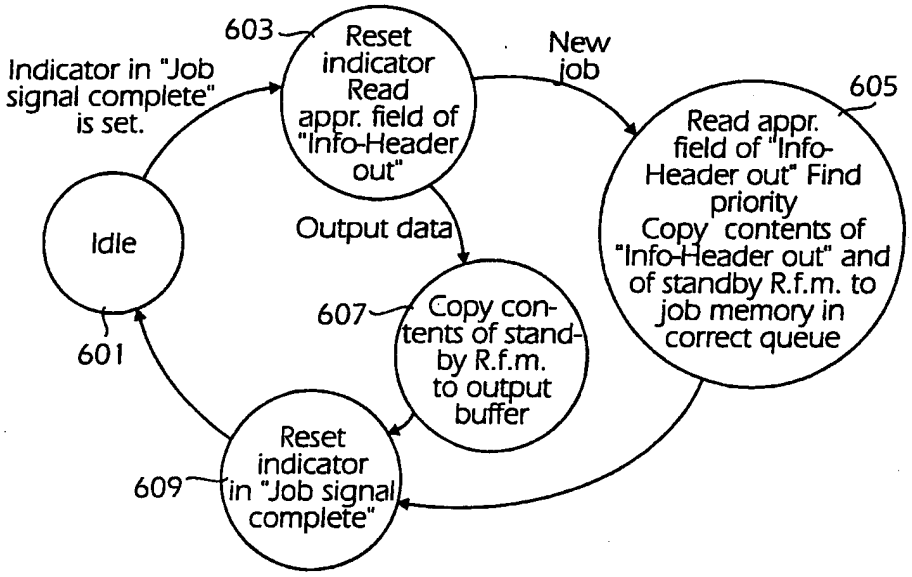

503
Wait
Job in job memory
505
Set indicator in "Busy" Read first job having highest priority, load fields Info and Header to "Info-Header In" and load data to to standby R.f.m.
Indicator in "Job read out complete" is set
Idle
501
Indicator in "Job taken" is set
511
Reset indicator in "Busy" Set indicator in "New job" Send interrupt signal
Reset indicator in "Busy" Set indicator in "New job" Wait
507
Job having having priority higher than that loaded to standby R.f.m. in job memory
Set indicator in "Busy" Reset indicator in "New job" Read fjob having higher priority, load fields Info and Header to "Info-Header In" and load data to to standby R.f.m.
509

Fig. 6


603
Reset indicator Read appr. field of "Info-Header out"
New job
605
Read appr. field of "Info-Header out" Find priority Copy contents of "Info-Header out" and of standby R.f.m. to job memory in correct queue
Indicator in "Job signal complete" is set.
Idle
601
Output data
607
Copy contents of stand-by R.f.m. to output buffer
609
Reset indicator in "Job signal complete"

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 99/01904

A. CLASSIFICATION OF SUBJECT MATTER

IPC7: G06F 9/46, G06F 9/38
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC7: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 9836355 A1 (THE DOW CHEMICAL COMPANY), 20 August 1998 (20.08.98), page 3, line 22 - page 4, line 31; page 5, line 22 - page 6, line 6; page 42, line 8 - line 19, figure 10A, 12A-12D, 15A --	1-5
A	EP 0405726 A2 (TEXAS INSTRUMENTS INCORPORATED), 2 January 1991 (02.01.91), figures 1a,1b, claims 1-3 --	1-5
A	JP 10207717 A (MATSUSHITA ELECTRIC IND CO LTD), 7 August 1998 (07.08.98), see whole document --	1-5

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" erlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
30 March 2000	05.04.2000
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Erik Veillas/CL Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 99/01904

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0696772 A2 (ROCKWELL INTERNATIONAL CORPORATION), 14 February 1996 (14.02.96), see whole document ---	1-5
A	US 5727211 A (DENIS GULSEN), 10 March 1998 (10.03.98), see whole document ---	1-5
A	US 5357617 A (GORDON T. DAVIS ET AL), 18 October 1994 (18.10.94), see whole document --- --------	1-5

INTERNATIONAL SEARCH REPORT

Information on patent family members

02/12/99

International application No.
PCT/SE 99/01904

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9836355 A1	20/08/98	AU 6263798 A	08/09/98
		US 5949994 A	07/09/99
EP 0405726 A2	02/01/91	DE 69032964 D,T	01/07/99
		EP 0884673 A	16/12/98
		EP 0884674 A	16/12/98
		EP 0892344 A	20/01/99
		JP 3063826 A	19/03/91
		JP 11096020 A	09/04/99
		JP 11096021 A	09/04/99
		US 5072418 A	10/12/91
		US 5535331 A	09/07/96
		US 5579218 A	26/11/96
		US 5579497 A	26/11/96
		US 5583767 A	10/12/96
		US 5586275 A	17/12/96
		US 5142677 A	25/08/92
		US 5313648 A	17/05/94
		US 5319789 A	07/06/94
		US 5319792 A	07/06/94
		US 5349687 A	20/09/94
		US 5550993 A	27/08/96
JP 10207717 A	07/08/98	NONE	
EP 0696772 A2	14/02/96	JP 8063361 A	08/03/96
		US 5655132 A	05/08/97
US 5727211 A	10/03/98	AU 7611296 A	29/05/97
		EP 0859978 A	26/08/98
		WO 9717654 A	15/05/97
US 5357617 A	18/10/94	JP 2500036 B	29/05/96
		JP 5224923 A	03/09/93